# Getting Started with Java DB

*Version 10.5*

Derby Document build:
May 8, 2009, 1:41:14 PM (EDT)

# Contents

Apache Software FoundationGetting Started with Java DBApache Derby

# Copyright



**Related information**

License

# License

### The Apache License, Version 2.0

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use,
      reproduction, and distribution as defined by Sections 1 through
      9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized
      by the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under
      common control with that entity. For the purposes of this
      definition, "control" means (i) the power, direct or indirect,
      to cause the direction or management of such entity, whether by
      contract or otherwise, or (ii) ownership of fifty percent (50%)
      or more of the outstanding shares, or (iii) beneficial ownership
      of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making
      modifications, including but not limited to software source code,
      documentation source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or
      Object form, that is based on (or derived from) the Work and
      for which the editorial revisions, annotations, elaborations,
      or other modifications represent, as a whole, an original work
      of authorship.  For the purposes of this License, Derivative
      Works shall not include works that remain separable from, or
      merely link (or bind by name) to the interfaces of, the Work
      and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or
      additions to that Work or Derivative Works thereof, that is
      intentionally submitted to Licensor for inclusion in the Work
      by the copyright owner or by an individual or Legal Entity
      authorized to submit on behalf of the copyright owner. For the
      purposes of this definition,
      "submitted" means any form of electronic, verbal, or written
      communication sent to the Licensor or its representatives,
      including but not limited to communication on electronic mailing
      lists, source code control systems, and issue tracking systems
```

# Relationship between Java(TM) DB and Derby

Java(TM) DB is a relational database management system that is based on the Java programming language and SQL. Java DB is a commercial release of the Apache Software Foundation's (ASF) open source relational database project. The Apache project is called Derby.

The Java DB product includes Derby without any modification whatsoever to the underlying source code.

Technical support is available for purchase for the Java DB product through Sun.

Because Java DB and Derby have the same functionality, the Java DB documentation refers to the core functionality as Derby.

Java DB Version 10.5 is based on the Derby Version 10.5 release. References to "Derby" in the Java DB documentation refer to the Version 10.5 release of Apache Derby.

# Introduction to Derby

Welcome to Derby! Derby is a full-featured, open source relational database management system (RDBMS) that is based on Java(TM) and SQL.

Derby is written and implemented completely in the Java programming language. Derby provides users with a small-footprint standards-based database engine that can be tightly embedded into any Java based solution. Derby ensures data integrity and provides sophisticated transaction support. In the default configuration there is no separate database server to be installed or maintained by the end user. For more information on Derby, visit the Derby Web site at http://db.apache.org/derby.

Before you install Derby, you should understand the deployment options and system requirements.

## Deployment options

The Derby software distribution provides two basic deployment options (also referred to as frameworks), the simple embedded option and the Derby Network Server option.

**Embedded**

Refers to Derby being started by a simple single-user Java application. With this option Derby runs in the same Java virtual machine (JVM) as the application. Derby can be almost invisible to the end user because it is started and stopped by the application and often requires no administration.

The Derby documentation often refers to this as the embedded configuration or embedded mode.

**Server (or Server-based)**

Refers to Derby being started by an application that provides multi-user connectivity to Derby databases across a network. With this option Derby runs in the Java virtual machine (JVM) that hosts the Server. Applications connect to the Server from different JVMs to access the database. The Derby Network Server is part of the Derby software distribution and provides this type of framework for Derby. Derby also works well with other, independently developed Server applications.

The Derby documentation often refers to this as the Network Server configuration or client/server configuration.

## System requirements

Derby is a database engine written completely in Java. The database will run in any certified Java Virtual Machine (JVM).

You must have a JAVA version 1.4 or higher installed on your computer. The Java Development Kit (JDK) is required to perform the activities in the Self-study tutorial for users new to Derby.

To check that the correct version of Java is installed, issue the `java -version` command.

## Product documentation for Java DB

The Java DB product documentation consists of the Java DB manuals and the API reference.

***Getting Started with Java DB***

Describes how to install and configure Java DB. Includes a self-study tutorial for users new to Java DB and a quick-start guide for experienced JDBC users. This guide introduces the `dblook`, `ij`, and `sysinfo` tools, and the libraries and scripts that are included with Java DB.

***Java DB Developer's Guide***

Describes the functionality and features of Java DB common to all deployments, such as Java DB's JDBC and SQL specifics, deploying Java DB applications, working with properties, security, and other advanced features.

***Java DB Reference Manual***

Documents the implementation of the SQL language in Java DB. This guide provides reference information about the JDBC and JTA implementations, keywords, system tables, properties, and *SQLExceptions* in Java DB.

***Tuning Java DB***

This guide offers performance tips, an in-depth discussion of performance, and information about the Java DB optimizer.

***Java DB Tools and Utilities Guide***

Describes how to use the the Java DB tools such as `dblook`, `ij`, and`sysinfo`, and how to use the system procedures to import and export data, and how to store Java code in the database.

***Java DB Server and Administration Guide***

Part One of this guide discusses configuring servers, how to program client applications, and database administration. In addition, some systems might require administrative tasks such as backing up databases. These tasks are independent of any server framework but are unique to multi-user or large systems.

Part Two of this guide discusses administrative issues such as backups and debugging deadlocks.

**Java DB API Reference**

An API Reference that is automatically generated for all public Java DB classes. No reference is provided for the JDBC API, which is part of the Java 2 Platform, Standard Edition. For more information about the classes in the API, see the *Java DB Reference Manual*.

You can access the Java DB manuals and API Reference from the Java DB Documentation page on the Java DB web site. The product documentation is also installed with Java DB. The manuals are installed in the `docs` subdirectory and the API Reference is installed in the `javadoc` subdirectory.

# Installing and configuring Derby

This section will help you install and configure Derby.

After you complete the installation:
- Access the Self-study tutorial to get up and running with Derby. This tutorial demonstrates how to use Derby in the embedded and client/server configurations.
- If you are an experienced JDBC programmer, you should also see the Quick start guide for experienced JDBC users.

## Installing Java DB

To install Java DB you must first download a Java DB install file from the Java DB Web site.

Navigate your Web browser to http://developers.sun.com/javadb/downloads/. Click Get It. Follow the instructions on subsequent pages.

Extract the downloaded package, which contains the distribution of Java DB. This distribution contains scripts, libraries, and demonstration programs. The extracted installation contains several subdirectories:
- The *bin* subdirectory contains the scripts for executing utilities and setting up the environment.
- The *demo* subdirectory contains the demonstration programs.
- The *docs* subdirectory contains the Java DB documentation.
- The *javadoc* subdirectory contains the `api` documentation that was generated from source code comments.
- The *lib* subdirectory contains the Java DB .jar files.

## Setting up your environment

Derby includes several scripts that run the Derby tools and utilities. How you run those scripts depends on the configuration that you want to use to access the Derby database and tools.

To set up your environment:
1. Choose the method that you want to use to run the Derby tools and utilities.
2. Set the environment variables.

### Choosing a method to run the Derby tools and startup utilities

There are several ways that you can run the Derby tools and startup utilities.

The method that you choose to run the tools and utilities determines the environment variables that you set for Derby.

1. Choose the method that you want to use:

| Method | When to Use | Requirements |
| --- | --- | --- |
| Run the tools using the command shell scripts that are provided with Derby. | Use this method when you want to run the tools with the least amount of typing and have installed the full `bin` distribution of Derby. | • Set the DERBY_HOME environment variable<br>• Include the `java.exe` file in the PATH environment variable |

| | | |
|---|---|---|
| | | • Include the `DERBY_HOME\bin` directory in the PATH environment variable |
| Run the tools using the `derbyrun.jar` archive file. | Use this method when:<br>• You have only the Derby jar files available (see Libraries provided by Derby), and do not have the full `bin` distribution of Derby<br>• You do not want to use the scripts that are provided with Derby | • Set the DERBY_HOME environment variable<br>• Include the `java.exe` file in the PATH environment variable<br><br>The `derbyrun.jar` file must be in the same folder as the other Derby .jar files.<br><br>For more information see the syntax for the `derbyrun.jar` file. |
| Run the tools using the complete `java` syntax or use Derby in a Java program. | Use this method when:<br>• You do not have the `derbyrun.jar` file or the Derby scripts installed.<br>• You want to learn the full syntax of each command and understand the details of setting up the execution environment.<br>• You are developing application programs with Derby | • Set the DERBY_HOME environment variable<br>• Include the `java.exe`file in the PATH environment variable<br>• You must know the full package name for the Java class that supports the tool<br>• The CLASSPATH environment variable must be set to include the required JAR files.<br><br>For details on setting the CLASSPATH, see Manually setting the CLASSPATH environment variable. |

2. Based on the requirements of the method that you chose to run the tools, follow the instructions to set the environment variables.

## Setting the environment variables

There are several environment variables that must be set depending on the method that you selected to run the Derby tools and startup utilities.

As mentioned in choosing a method to run the Derby tools and startup utilities, you must set the DERBY_HOME environment variable so that you can use the command examples that are presented in this manual. Adding the Derby scripts directory to your command execution PATH makes the scripts easier to use and enables you to use the script examples in this manual. The CLASSPATH environment variable must be set if you are using Derby in a Java program or executing the tools using the `java` command. The steps below show you how to set the environment variables in a command window. The settings are only valid for that window. If you close the command window or open a new command window, you must set the environment variables again.
**Tip:** You can also set environment variables permanently. For example, on Windows you can use the Control Panel to permanently set the environment variables.

To set the environment variables:

1. Set the DERBY_HOME environment variable to the location where you extracted the Derby *bin* distribution.

For example, if you installed Derby in the `/opt/Derby_10` directory on UNIX or in the `c:\Derby_10` directory on Windows, use the following command to set the DERBY_HOME environment variable:

| Operating System | Command |
|---|---|
| **UNIX** | **`export DERBY_HOME=/opt/Derby_10`** |
| **Windows** | **`set DERBY_HOME=c:\Derby_10`** |

2. Be certain that the `java.exe` file, version 1.4.2 or, higher is in your command execution PATH. Open a command window and run the `java -version` command.
3. Add the `DERBY_HOME/bin` directory to the PATH environment variable so that you can run the Derby scripts from any directory.

| Operating System | Command |
|---|---|
| **UNIX** | **`export PATH="$DERBY_HOME/bin:$PATH"`** |
| **Windows** | **`set PATH=%DERBY_HOME%\bin;%PATH%`. If you use the Control Panel to update your system PATH, add `%DERBY_HOME%\bin` to the end of the PATH environment variable** |

**Tip:** When the DERBY_HOME environment variable is set and the underlying `/bin` directory is included in the PATH environment variable, you can use shortened commands to start the Derby tools. Otherwise, either you must be in the directory where the script that starts the Derby tool is located, or you must specify the full path to the location of the script when you want to start the tool.

For more information on the scripts included in the *bin* distribution, see Scripts included with Derby.

## Syntax for the derbyrun.jar file

The `derbyrun.jar` file is a special JAR file that simplifies how you invoke the Derby tools and the Network Server.

With the `derbyrun.jar` file, you can run the Derby tools and utilities using shortened names and you do not need to set the java CLASSPATH environment variable. The `derbyrun.jar` file must be in the same folder as the other Derby JAR files.

**Syntax**

The syntax for using `derbyrun.jar` for each of the Derby tools is as follows:

| Operating System | Command |
|---|---|
| UNIX (Korn Shell) | `java -jar $DERBY_HOME/lib/derbyrun.jar ij [-p propertiesfile] [sql_script]`<br>`java -jar $DERBY_HOME/lib/derbyrun.jar sysinfo [-cp ...] [-cp help]`<br><br>`java -jar $DERBY_HOME/lib/derbyrun.jar dblook [arg]* (or no arguments for usage)`<br>`java -jar $DERBY_HOME/lib/derbyrun.jar server [arg]* (or no arguments for usage)` |
| Windows | `java -jar %DERBY_HOME%\lib\derbyrun.jar ij [-p propertiesfile] [sql_script]` |

| | |
|---|---|
| | `java -jar %DERBY_HOME%\lib\derbyrun.jar sysinfo [-cp ...] [-cp help]`<br>`java -jar %DERBY_HOME%\lib\derbyrun.jar dblook [arg]* (or no arguments for usage)`<br>`java -jar %DERBY_HOME%\lib\derbyrun.jar server [arg]* (or no arguments for usage)` |

To see this syntax reminder, run the command `java -jar derbyrun.jar` with no arguments.

**Additional information**

The `server` argument is a shortcut for running the `NetworkServerControl` tool. For details on using this tool, see the *Java DB Server and Administration Guide*.

For more information on using `derbyrun.jar` to run the `ij`, `sysinfo`, and `dblook` tools, see the *Java DB Tools and Utilities Guide*.

## Manually setting the CLASSPATH environment variable

You can set the CLASSPATH environment variable in the operating system either temporarily, permanently, or at run time when you start your Java application and the JVM.

The classpath is a list of the class libraries that are needed by the JVM and other Java applications to run your program. There are scripts that are included with Derby that can set up the classpath to run the Derby tools.

If you want to call the Derby tools directly using Java and not using the scripts, you must manually set the CLASSPATH environment variable.

In most development environments, it is best to temporarily set the CLASSPATH environment variable in the command line shell where you are entering commands.

Derby provides several scripts in the DERBY_HOME`/bin` directory to help you set your classpath quickly. These scripts are:

**setEmbeddedCP**
: Use the setEmbeddedCP script to set the classpath when the database engine is used in embedded mode. This script adds the `derby.jar` and `derbytools.jar` files to the classpath.

**setNetworkServerCP**
: Use the setNetworkServerCP script to set the classpath when you want to start the network server. This script adds the `derbynet.jar` file to the classpath.

**setNetworkClientCP**
: Use the setNetworkClientCP script to set the classpath when you want to access databases using the network client. This script adds the `derbyclient.jar` and `derbytools.jar` files to the classpath.

To set the classpath temporarily, run the script that is appropriate for your environment every time that you open a new command window.

To see the classpath that the script sets, issue the following command after you run the script:

- On UNIX, use `echo $CLASSPATH`
- On Windows, use `echo %CLASSPATH%`

For more information on running the `ij` and `sysinfo` utilities, see the *Java DB Tools and Utilities Guide*

# Using the Derby tools and startup utilities

There are several tools and utilities that you might want to use as you begin to work with Derby.

The tools that are included with Derby are `dblook`, `ij`, and `sysinfo`. To run these tools see:

- Running sysinfo
- Running ij
- Running dblook

Derby includes a set of scripts that you can use to start the Derby tools. The scripts are located in the DERBY_HOME/`bin` directory. When you run these scripts, your CLASSPATH environment variable is set if you have not already set it, and remains set as long as you are running the tool.

Most of the examples in this guide that show how to use the Derby scripts to launch the Derby tools assume that you are using the embedded mode of the Derby database engine. Use the instructions below to run the scripts with the Network Server.

**Running the scripts with the Network Server**

To run the scripts with the Network Server, use the following commands:

- For the `sysinfo` tool, issue the command

  ```
  NetworkServerControl sysinfo
  ```
- For the `ij` tool, issue the command

  ```
  set DERBY_OPTS=-Dij.protocol=jdbc:derby://localhost/
  ```

  and then start `ij` by issuing the command `ij`.
- For the `dblook` tool, call the script and specify the `-d` option and the full connection URL to the Network Server database. For example:

  ```
  dblook -d 'jdbc:derby://localhost/myDB;user=usr'
  ```

**Additional Derby utilities**

In addition, there are Derby utilities that are system procedures that you can call by using the `ij` tool. For example, there are system procedures that you can use to import and export external files. Instructions on how to use these system procedures are included in the *Java DB Tools and Utilities Guide* and the *Java DB Reference Manual*.

## Running sysinfo

The Derby `sysinfo` tool displays information about your Java environment and your version of Derby.

The `sysinfo` utility prints system information to a console.

Choose the method that you can use to run the `sysinfo` script:

| Method | When to Use | Command |
| --- | --- | --- |
| Run `sysinfo` as a standalone command. | Use this method if you are relatively new to the Java programming language and new to Derby. | You must set your environment variables before you can run the `sysinfo` utility using this method.<br>To run the `sysinfo` script from the command line us:<br>`sysinfo` |

| | | |
|---|---|---|
| | | The `sysinfo` script sets the appropriate environment variables, including the CLASSPATH, and runs the `sysinfo` utility. |
| Run `sysinfo` using the jar file that is located in the directory where `sysinfo` resides. | Use this method if you are new to Derby, but are familiar with the Java programming language. | You must set the DERBY_HOME environment variable before you can run the `sysinfo` utility using this method.<br>On UNIX, the command is:<br>`java [options] -jar $DERBY_HOME/lib/derbyrun.jar sysinfo`<br>On Windows, the command is:<br>`java [options] -jar %DERBY_HOME%\lib\derbyrun.jar sysinfo` |
| Run `sysinfo` using the `java` command. | Use this method if you are familiar with both the Java programming language and Derby, and you have already set the `java.exe` file in your command execution PATH. | You must set your CLASSPATH. Use the steps specified in Manually setting the CLASSPATH environment variable. Then specify the class name in the `java` command. For example:<br>`java org.apache.derby.tools.sysinfo` |

See "sysinfo" in the *Java DB Tools and Utilities Guide* for more information about using the `sysinfo` utility.

## Running ij

The Derby `ij` tool is a JDBC tool that you can use to run scripts or interactive queries against a Derby database.

- Choose the method that you can use to run the `ij` script:

| Method | When to Use | Command |
|---|---|---|
| Run `ij` as a standalone command. | Use this method if you are relatively new to the Java programming language and new to Derby. | You must set your environment variables before you can run the `ij` tool using this method. To run the `ij` script from the command line use:<br>`ij`<br>You must add the DERBY_HOME`/bin` directory in your PATH environment variable before you can run the `ij` tool.<br>The `ij` script sets up the environment variables like CLASSPATH and starts the `ij` tool. |
| Run `ij` using the jar file that is located in the directory where `ij` resides. | Use this method if you are new to Derby, but are familiar with the | You must set the DERBY_HOME environment variable before you can run the `ij` tool using this method.<br>On UNIX, the command is: |

| | | |
|---|---|---|
| | Java programming language. | `java -jar $DERBY_HOME/lib/derbyrun.jar ij`<br><br>On Windows, the command is:<br><br>`java -jar %DERBY_HOME%\lib\derbyrun.jar ij` |
| Run `ij` using the `java` command. | Use this method if you are familiar with both the Java programming language and Derby, and you have already set the `java.exe` file in your command execution PATH. | You must set your CLASSPATH. Use the steps specified in Manually setting the CLASSPATH environment variable. Then specify the class name in the `java` command. For example:<br><br>`java org.apache.derby.tools.ij` |

- When you are ready to leave the `ij` tool, type:

```
ij> exit;
```

See the *Java DB Tools and Utilities Guide* for more information about the `ij` tool.

## Running dblook

The Derby `dblook` utility is a Data Definition Language (DDL) generation utility.

The `dblook` utility is a simple utility that dumps all or parts of the DDL of a user-specified database to either a console or a file. The generated DDL can then be used for such things as recreating all or parts of a database, viewing a subset of the objects in a database (for example, those objects that pertain to specific tables and schemas), or documenting the schema of a database.

Choose the method that you can use to run the `dblook` script:

| Method | When to Use | Command |
|---|---|---|
| Run `dblook` as a standalone command. | Use this method if you are relatively new to the Java programming language and new to Derby. | You must set your environment variables before you can run the `dblook` utility using this method.<br><br>To run the `dblook` script from the command line use:<br><br>`dblook -d` *`connectionURL`* `[`*`options`*`]`<br><br>The `dblook` script sets the appropriate environment variables, including the CLASSPATH, and runs the `dblook` utility. |
| Run `dblook` using the jar file that is located in the directory where `dblook` resides. | Use this method if you are new to Derby, but are familiar with the Java programming language. | You must set the DERBY_HOME environment variable before you can run the `dblook` utility using this method.<br><br>On UNIX, the command is:<br><br>`java [options] -jar $DERBY_HOME/lib/derbyrun.jar dblook -d` *`connectionURL`* `[`*`options`*`]` |

| | | |
|---|---|---|
| | | On Windows, the command is:<br><br>`java [options] -jar %DERBY_HOME%\lib\derbyrun.jar dblook -d connectionURL [options]` |
| Run `dblook` using the `java` command. | Use this method if you are familiar with both the Java programming language and Derby, and you have already set your command execution PATH to the location of your `java` command. | You must set your CLASSPATH. Use the steps specified in Manually setting the CLASSPATH environment variable. Then specify the class name in the `java` command. For example:<br><br>`java org.apache.derby.tools.dblook -d connectionURL [options]` |

See Using the Derby tools and startup utilities for an example of the use of the *connectionURL*. See "dblook" and its subtopics in the *Java DB Tools and Utilities Guide* for more information about using the `dblook` utility.

# Verifying the Derby system configuration

You should confirm that your computer has the correct software installed and that the environment variables are set before you start working with Derby.

To check the Derby system configuration:

1. Verify that the `java.exe` file, version 1.4.2 or, higher is in your command execution PATH by opening a command window and running the `java -version` command.

   The output from the command looks something like this:

   ```
   java version "1.4.2_04"
   Java(TM) 2 Runtime Environment, Standard Edition (build
    1.4.2_04-b05)
   Java HotSpot(TM) Client VM (build 1.4.2_04-b05, mixed mode)
   ```

   The output you see might be different from what is shown here because the `java -version` command outputs vendor-specific information. If the command produces an error or the version listed is not 1.4 or higher, install a JDK before continuing.
2. Verify that the DERBY_HOME environment variable is set and points to the root directory of the Derby 10.5 installation. Open a command window and run the appropriate command for your system.

   If you installed Derby in the /opt/Derby_10 directory on UNIX or the c:\Derby_10 directory on Windows, the command that you use and the expected output are shown in the following table:

   | Operating System | Command | Output |
   |---|---|---|
   | UNIX (Korn Shell) | `echo $DERBY_HOME` | `/opt/Derby_10` |
   | Windows | `echo %DERBY_HOME%` | `C:\Derby_10` |

   If you receive an error or do not see the expected output, ensure that you have set the `DERBY_HOME` environment variable, as described in Setting the environment variables.

**Note:** You can also use the `sysinfo` utility to verify that the environment variables are set correctly. If the environment variables are set correctly, the `sysinfo` command

displays information about your Java virtual machine (JVM) and the version of Derby that you have installed.

# Self-study tutorial for users new to Derby

To help you get up and running with Derby as quickly as possible, this self-study guide highlights some of the more important features of Derby.

This tutorial uses a series of activities designed to demonstrate the use of Derby in the embedded and client/server configurations. After performing these activities, you will find Derby to be an easy-to-use and fully functional relational database management system (RDBMS).

**Tutorial skills and software requirements**

To perform the tutorial activities, you do not need to have any prior knowledge of Java software, the Java Database Connectivity (JDBC) API, or SQL. Each activity provides the complete command syntax that you need to use for each operation. Instructions include the syntax for both a UNIX Korn shell and a Windows command prompt.

This tutorial demonstrates, but does not teach, the Java, JDBC and SQL code that is presented. If you want a deeper understanding of these topics, you will need to consult additional reference materials.

To perform the tutorial activities, you must have:

- A Java Development Kit (JDK) version 1.4 or higher installed on your computer
- The binary (`bin`) installation of Apache Derby version 10.5 installed on your computer
- A basic knowledge of the computer command line interface:
  - How to start a command shell or window
  - How to navigate the file system hierarchy

**System configuration**

You should verify the system configuration before you perform the tutorial activities.

**Problems and feedback on the tutorial activities**

If you experience problems with any aspect of the tutorial activities, send an e-mail message to `derby-user@db.apache.org`. Someone from the Derby community will respond to your message.

The questions and feedback received will be used to make this tutorial more useful.

## Tutorial overview

Each activity in this self-study tutorial highlights an important feature of Derby.

Here is what you can expect to learn in each activity:

**Activity 1:**
Use the Derby `ij` tool to load the Derby embedded driver and start the Derby database engine. Create the database `firstdb` and the table `FIRSTTABLE`. Use a few basic SQL statements to insert and select data. The Derby message log `derby.log` and the database directories and files are introduced.

**Activity 2:**
Use Derby within a client/server configuration. Start the Derby Network Server, which will embed the Derby engine. In a separate process, use the Derby `ij` tool to load the Derby client driver and connect to the Network Server. Create a database called `seconddb` and the table `SECONDTABLE`. Use a few basic SQL statements to insert and select data.

**Activity 3:**

Load the Derby database engine from a simple Java JDBC program. Use the embedded driver to create the database `jdbcDemoDB` and the `WISH_LIST` table. Populate the table with text entered from the keyboard, then view a list of the records in the table. Walk through the code to understand the basic structure of a JDBC program that accesses a Derby database. The `CLASSPATH` variable and connection URL attribute `;shutdown=true` are introduced.

**Activity 4:**

Modify the Java JDBC program to load the client driver and connect to the Derby Network Server. Compile the altered program and test that the program operates as it did in the previous activity.

# Activity 1: Run SQL using the embedded driver

In this activity, you will use the embedded driver to load the Derby database engine, create and connect to a database, and use some basic SQL statements.

To run SQL using the embedded driver:

1. Create a directory and copy scripts into the directory
2. Create a Derby database and run SQL statements

## Creating a directory and copying scripts into the directory

To help you complete this activity, you must create a directory and copy several scripts into the new directory. You will use these scripts to quickly add tables and data to a new Derby database.

**Tip:** A command prompt appears after each command is run. If an error occurs, verify the syntax and retype the command.

1. Open a command window and change to a directory where you want to store the files that you create during the self-study tutorial activities.
2. Create the `DERBYTUTOR` directory. `DERBYTUTOR` will be your working directory for this activity.

| Operating System | Command |
|---|---|
| UNIX (Korn Shell) | `mkdir DERBYTUTOR` |
| Windows | `md DERBYTUTOR` |

3. Change to the `DERBYTUTOR` directory.

| Operating System | Command |
|---|---|
| UNIX (Korn Shell) | `cd DERBYTUTOR` |
| Windows | `cd DERBYTUTOR` |

4. Copy the SQL scripts from the Derby `demo\programs\toursdb` subdirectory into the `DERBYTUTOR` directory. You will use these scripts to create tables and add data to a new database, `toursdb`.

| Operating System | Command |
|---|---|
| UNIX | `cp $DERBY_HOME/demo/programs/toursdb/*.sql .` |

| | |
|---|---|
| (Korn Shell) | |
| Windows | `copy %DERBY_HOME%\demo\programs\toursdb\*.sql .` |

5. Verify that the files were copied to the DERBYTUTOR directory.

| Operating System | Command |
|---|---|
| UNIX (Korn Shell) | `ls DERBYTUTOR` |
| Windows | `dir DERBYTUTOR` |

**> Important:** Include the dot (.) at the end of each command so that your current working directory is included in the classpath and the files are copied to the correct location.

## Creating a Derby database and running SQL statements

Now, you will use the Derby ij tool to load the Derby database engine. You will use the Derby embedded driver to create and connect to the firstdb database. You will also use a few basic SQL statements to create and populate a table.

1. Run the Derby ij tool.

   If you included the DERBY_HOME/bin directory in your PATH environment variable, type:

   ```
   ij
   ```

   Otherwise, you can use the java command to start the ij tool.

| Operating System | Command |
|---|---|
| UNIX (Korn Shell) | `java -jar $DERBY_HOME/lib/derbyrun.jar ij`<br>`ij version 10.5` |
| Windows | `java -jar %DERBY_HOME%\lib\derbyrun.jar ij`<br>`ij version 10.5` |

2. Create the database and open a connection to the database using the embedded driver.

   ```
   CONNECT 'jdbc:derby:firstdb;create=true';
   ```

   **Description of connection command:**

   **connect**
   The ij command to establish a connection to a database. The Derby connection URL is enclosed in single quotation marks. An ij command can be in either uppercase or lowercase.

   **jdbc:derby:**
   The JDBC protocol specification for the Derby driver.

   **firstdb**
   The name of the database. The name can be any string. Because no filepath is specified, the database is created in the default working directory (DERBYTUTOR).

   **;create=true**
   The Derby *URL attribute* that is used to create a database. Derby does not have an SQL create database command.

   **;**

The semicolon is the `ij` command terminator.

3. Create a table with two columns using standard SQL.

```
CREATE TABLE FIRSTTABLE
    (ID INT PRIMARY KEY,
    NAME VARCHAR(12));
0 rows inserted/updated/deleted
```

4. Insert three records.

```
INSERT INTO FIRSTTABLE VALUES
    (10,'TEN'),(20,'TWENTY'),(30,'THIRTY');
3 rows inserted/updated/deleted
```

5. Perform a simple select of all records in the table.

```
SELECT * FROM FIRSTTABLE;
ID         |NAME
------------------------
10         |TEN
20         |TWENTY
30         |THIRTY

3 rows selected
```

6. Perform a qualified select of the record with column ID=20.

```
SELECT * FROM FIRSTTABLE
WHERE ID=20;
ID         |NAME
------------------------
20         |TWENTY

1 row selected
```

7. Optional: Create and populate additional tables and other schema objects.

    a. Load the SQL script `ToursDB_schema.sql`.

    ```
    run 'ToursDB_schema.sql';

    ij> ...
    CREATE TABLE AIRLINES
       (
          AIRLINE CHAR(2) NOT NULL ,
          AIRLINE_FULL VARCHAR(24),
          BASIC_RATE DOUBLE PRECISION,
       ...
    0 rows inserted/updated/deleted
       ... Other output messages not shown ...
    ```

    b. Populate the tables with data by running the script `loadTables.sql`.

    ```
    run 'loadTables.sql';

    ij> run 'loadCOUNTRIES.sql';
    ij> insert into COUNTRIES values ( 'Afghanistan','AF','Asia');
    1 row inserted/updated/deleted
    ij> insert into COUNTRIES values ( 'Albania','AL','Europe');
    1 row inserted/updated/deleted
       ... Other output messages not shown ...
    ```

8. Exit the `ij` tool.

```
exit;
```

You should be returned to the `DERBYTUTOR` directory.

9. Browse the most important files created by this activity:

    - The `derby.log` file. This file is a message and error log that, under normal circumstances, contains a set of startup messages and a shutdown message.

```
------------------------------------------------------------------
2009-05-08 16:55:29.736 GMT:
 Booting Derby version The Apache Software Foundation - Apache
  Derby - 10.5.1.1 - (764942):
  instance a816c00e-0121-2126-a56a-000000418588
  on database directory C:\DERBYTUTOR\firstdb

Database Class Loader started - derby.database.classpath=''

2009-05-08 16:56:37.367 GMT:
Shutting down instance a816c00e-0121-2126-a56a-000000418588
------------------------------------------------------------------
```

- The `firstdb` database directory. Within the directory are the subdirectories `seg0` (containing the data files) and `log` (containing the transaction log files).

## Activity 2: Run SQL using the client driver

This activity uses Derby within a client/server configuration by using the Network Server. The `ij` tool is the client application that connects to the Derby Network Server. In this activity, you create a database called `seconddb` and run some basic SQL statements.

This activity assumes that you know how to open a command shell and change to the `DERBYTUTOR` directory.

You use two command windows (referred to as Shell-1 and Shell-2) in this activity. You use Shell-1 to start the Derby Network Server and display Network Server messages. You use Shell-2 to establish a client connection to the Network Server using `ij` and then perform some basic SQL operations.

1. Open a command window (Shell-1) and change to the `DERBYTUTOR` directory.
2. Start the Network Server.

| Operating System | Command |
|---|---|
| UNIX (Korn Shell) | **java -jar $DERBY_HOME/lib/derbyrun.jar server start**<br>2009-05-08 17:00:50.077 GMT : Security manager installed using the Basic server security policy.<br>2009-05-08 17:00:50.798 GMT : Apache Derby Network Server - 10.5.1.1 - (764942) started and ready to accept connections on port 1527 |
| Windows | **java -jar %DERBY_HOME%\lib\derbyrun.jar server start**<br>2009-05-08 17:00:50.077 GMT : Security manager installed using the Basic server security policy.<br>2009-05-08 17:00:50.798 GMT : Apache Derby Network Server - 10.5.1.1 - (764942) started and ready to accept connections on port 1527 |

   A Network Server startup message appears in the Shell-1 command window.
3. Open another command window (Shell-2). Change to the `DERBYTUTOR` directory.
4. Start `ij`.

   If you included the `DERBY_HOME/bin` directory in your PATH environment variable, type:

```
ij
```

   Otherwise, you can use the `java` command to start the `ij` tool.

| | Command |
|---|---|

| Operating System | |
|---|---|
| UNIX (Korn Shell) | **java -jar $DERBY_HOME/lib/derbyrun.jar ij**<br>ij version 10.5 |
| Windows | **java -jar %DERBY_HOME%\lib\derbyrun.jar ij**<br>ij version 10.5 |

You will enter all subsequent commands from the network client, so you will type the commands in the Shell-2 command window.

5. Create and open a connection to the database using the client driver.

```
CONNECT 'jdbc:derby://localhost:1527/seconddb;create=true';
```

**Remember:** A client connection URL contains a hostname and a port number. For example: `//localhost:1527/`

6. Create a table with two columns (ID and NAME) using the following SQL statement:

```
CREATE TABLE SECONDTABLE
    (ID INT PRIMARY KEY,
    NAME VARCHAR(14));
0 rows inserted/updated/deleted
```

7. Insert three records into the table.

```
INSERT INTO SECONDTABLE VALUES
    (100,'ONE HUNDRED'),(200,'TWO HUNDRED'),(300,'THREE HUNDRED');
3 rows inserted/updated/deleted
```

8. Select all of the records in the table.

```
SELECT * FROM SECONDTABLE;
ID         |NAME
------------------------
100        |ONE HUNDRED
200        |TWO HUNDRED
300        |THREE HUNDRED

3 rows selected
```

9. Select a subset of records from the table by specifying a `WHERE` clause.

```
SELECT * FROM SECONDTABLE WHERE ID=200;
ID         |NAME
------------------------
200        |TWO HUNDRED

1 row selected
```

10. Exit `ij`.

```
exit;
```

11. Shut down the Derby Network Server.

| Operating System | Command |
|---|---|
| UNIX (Korn Shell) | **java -jar $DERBY_HOME/lib/derbyrun.jar server shutdown**<br>2009-05-08 17:07:25.887 GMT : Apache Derby Network Server - 10.5.1.1 - (764942) shutdown |
| Windows | **java -jar %DERBY_HOME%\lib\derbyrun.jar server shutdown**<br>2009-05-08 17:07:25.887 GMT : Apache Derby Network Server - 10.5.1.1 - (764942) shutdown |

The server shutdown confirmation appears in both command windows.

**Activity notes**

The client connection URL contains network information (hostname and portnumber) not found in the URL for an embedded connection. This information tells the client driver the location of the Network Server. The client driver sends requests to and receives responses from the Network Server.

In this activity the Derby database engine is embedded in the Network Server and returns data to the `ij` client (a client/server configuration). In contrast, establishing a connection using an embedded URL (one without `//localhost:1527/`) would have caused the Derby engine to be embedded in the `ij` application (an embedded configuration).

In this configuration, multiple client programs can connect to the Network Server and access the database simultaneously. (This activity does not demonstrate this capability.)

## Activity 3: Run a JDBC program using the embedded driver

This activity loads the Derby database engine using a simple Java JDBC program.

This activity assumes that you have opened a command window and navigated to the `DERBYTUTOR` directory.

JDBC is the Java Database Connectivity API and is also the native API for Derby. The program uses the embedded driver to create the `jdbcDemoDB` database (if the database does not exist) and then connect to the database. You can then populate a table within the database with text. The program demonstrates some basic JDBC processing along with related error handling.

The Java compiler and runtime use the classpath, specified by the `CLASSPATH` environment variable, to locate the binary files (jar files and class files) that are needed to run Derby and other Java applications. Before performing this activity, you need to set the classpath and compile the `WwdEmbedded.java` program.

1. Copy the program files into the `DERBYTUTOR` directory and set the CLASSPATH environment variable.

| Operating System | Commands |
| --- | --- |
| UNIX (Korn Shell) | **`cp $DERBY_HOME/demo/programs/workingwithderby/* .`**<br>**`export CLASSPATH=$DERBY_HOME/lib/derby.jar:.`** |
| Windows | **`copy %DERBY_HOME%\demo\programs\workingwithderby\* .`**<br>**`set CLASSPATH=%DERBY_HOME%\lib\derby.jar;.`** |

   **> Important:** Include the dot (`.`) at the end of each command so that your current working directory is included in the classpath and the files are copied to the correct location.
2. Compile the `WwdEmbedded.java` program.

```
javac WwdEmbedded.java
```

   **> Important:** A command prompt appears if the compilation is successful. The binary file `WwdEmbedded.class` is created. If an error message appears, verify that the JDK is properly installed.
3. Run the program.

The WwdEmbedded.java program populates a table with wish-list items. The program prompts you for text input (up to 32 characters), stores the text input in a database table, and then lists the items stored in the table. The program continues to ask for wish-list items until the you type the command `exit` or a problem is encountered. Some basic information on program progress is displayed at the beginning and the end of the program.

```
java WwdEmbedded
org.apache.derby.jdbc.EmbeddedDriver loaded.
Connected to database jdbcDemoDB
 . . . . creating table WISH_LIST
Enter wish-list item (enter exit to end):
a peppermint stick
  __________________________________________________________
On 2009-05-08 13:12:09.058 I wished for a peppermint stick
  __________________________________________________________
Enter wish-list item (enter exit to end):
a long vacation
  __________________________________________________________
On 2009-05-08 13:12:09.058 I wished for a peppermint stick
On 2009-05-08 13:12:21.28 I wished for a long vacation
  __________________________________________________________
Enter wish-list item (enter exit to end):
exit
Closed connection
Database shut down normally
Getting Started With Derby JDBC program ending.
```

## The WwdEmbedded program

This section describes the `WwdEmbedded.java` program, highlighting details specific to accessing a Derby database from a JDBC program.

Most of the code related to the database activities performed is included in this section, but you might find it helpful to open the program file and follow along in a text viewer or editor. The *SECTION NAMES* referred to in this section can be found in the comments within the program code and serve as cross-reference points between this section and the Java program. The program uses methods from the `WwdUtils` class. The utility class code is not described here but is available for review in the file `WwdUtils.java`.

### Initialize the program

*INITIALIZATION SECTION*: The initial lines of code identify the Java packages used in the program, then set up the Java class `WwdEmbedded` and the `main` method signature. Refer to a Java programming guide for information on these program constructs.

```
import java.sql.*;

public class WwdEmbedded
{
    public static void main(String[] args)
   {
```

### Define key variables and objects

*DEFINE VARIABLES SECTION*: The initial lines of the `main` method define the variables and objects used in the program. This example uses variables to store the information needed to connect to the Derby database. The use of variables for this information makes it easy to adapt the program to other configurations and other databases.

**`driver`**
Stores the name of the Derby embedded driver.

**`dbName`**
Stores the name of the database.

**`connectionURL`**
Stores the Derby connection URL that is used to access the database.

**`createString`**
Stores the SQL `CREATE` statement for the `WISH_LIST` table.

```
String driver = "org.apache.derby.jdbc.EmbeddedDriver";
String dbName="jdbcDemoDB";
String connectionURL = "jdbc:derby:" + dbName + ";create=true";
...
String createString = "CREATE TABLE WISH_LIST  "
  +  "(WISH_ID INT NOT NULL GENERATED ALWAYS AS IDENTITY "
  ...
  +  " WISH_ITEM VARCHAR(32) NOT NULL) " ;
```

### Start the Derby engine

*LOAD DRIVER SECTION*: Loading the Derby embedded JDBC driver starts the Derby database engine. The `try` and `catch` block (the Java error-handling construct) catches the exceptions that may occur. A problem here is usually due to an incorrect classpath setting.

```
String driver = "org.apache.derby.jdbc.EmbeddedDriver";
...
try {
    Class.forName(driver);
} catch(java.lang.ClassNotFoundException e) {
  ...
}
```

If you are using JDK 6 or higher, this code fragment is not needed, because you do not need to load the driver explicitly. In a JDK 6 environment, the driver loads automatically when you obtain a connection to the database.

### Boot the database

*BOOT DATABASE SECTION*: The `DriverManager` class loads the database using the Derby connection URL stored in the variable `connectionURL`. This URL includes the parameter `;create=true` so that the database will be created if it does not already exist. The primary `try` and `catch` block begins here. This construct handles errors for the database access code.

```
String connectionURL = "jdbc:derby:" + dbName + ";create=true";
...
try {
    conn = DriverManager.getConnection(connectionURL);
    ...  <most of the program code is contained here>
}  catch (Throwable e)  {
   ...
}
```

### Set up program to execute SQL

*INITIAL SQL SECTION*: The program initializes the objects needed to perform subsequent SQL operations and checks to see if the required data table exists.

The statement object `s` is initialized. If the utility method `WwdUtils.wwdChk4Table` does not find the `WISH_LIST` table, the statement object's `execute` method creates the table by executing the SQL stored in the variable `createString`.

```
s = conn.createStatement();
if (! WwdUtils.wwdChk4Table(conn))
{
   System.out.println (" . . . . creating table WISH_LIST");
   s.execute(createString);
}
```

The `INSERT` statement used to add data to the table is bound to the prepared statement object `psInsert`. The prepared statement uses the question mark parameter ? to represent the data that will be inserted by the user. The program sets the actual value to be inserted later on, before executing the SQL. This is the most efficient way to execute SQL statements that will be used multiple times.

```
psInsert = conn.prepareStatement
   ("insert into WISH_LIST(WISH_ITEM) values (?)");
```

**Interact with the database**

*ADD / DISPLAY RECORD SECTION*: This section uses the utility method `WwdUtils.getWishItem` to gather information from the user. It then uses the objects set up previously to insert the data into the `WISH_LIST` table and then display all records. A standard do loop causes the program to repeat this series of steps until the user types `exit`. The data-related activities performed in this section are as follows:

- The `setString` method sets the substitution parameter of the `psInsert` object to the value typed by the user. Then the `executeUpdate` method performs the database insert.

  ```
  psInsert.setString(1,answer);
  psInsert.executeUpdate();
  ```
- The statement object `s` is used to select all the records in the `WISH_LIST` table and store them in the `ResultSet` named `myWishes`.

  ```
  myWishes = s.executeQuery("select ENTRY_DATE, WISH_ITEM
                 from WISH_LIST order by ENTRY_DATE");
  ```

  The `while` loop reads each record in turn by calling the `next` method. The `getTimestamp` and `getString` methods return specific fields in the record in the proper format. The fields are displayed using rudimentary formatting.

  ```
  while (myWishes.next())
  {
      System.out.println("On " + myWishes.getTimestamp(1) +
         " I wished for " + myWishes.getString(2));
  }
  ```

  Close the `ResultSet` to release the memory being used.

  ```
  myWishes.close();
  ```

**Shut down the database**

*DATABASE SHUTDOWN SECTION*: If an application starts the Derby engine, the application should shut down all databases before exiting. The attribute `;shutdown=true` in the Derby connection URL performs the shutdown. When the Derby engine is shutdown, all booted databases will automatically shut down. The shutdown process cleans up records in the transaction log to ensure a faster startup the next time the database is booted.

**Tip:** You can shut down individual databases without shutting down the engine by including the database name in the connection URL.

"This section verifies that the embedded driver is being used, then issues the shutdown command and catches the shutdown exception to confirm that the Derby engine shut down cleanly. The shutdown status is displayed before the program exits.

```
if (driver.equals("org.apache.derby.jdbc.EmbeddedDriver")) {
   boolean gotSQLExc = false;
   try {
      DriverManager.getConnection("jdbc:derby:;shutdown=true");
   } catch (SQLException se)  {
```

```
      if ( se.getSQLState().equals("XJ015") ) {
         gotSQLExc = true;
      }
   }
   if (!gotSQLExc) {
      System.out.println("Database did not shut down normally");
   }  else  {
      System.out.println("Database shut down normally");
   }
}
```

**> Important:** The XJ015 error (successful shutdown of the Derby engine) and the 08006 error (successful shutdown of a single database) are among the few exceptions thrown by Derby that indicate that an operation succeeded. Most other exceptions indicate that an operation failed. You should check the log file to be certain.

**The errorPrint and SQLExceptionPrint methods**

*DERBY EXCEPTION REPORTING CLASSES*: The two methods at the end of the file, `errorPrint` and `SQLExceptionPrint`, are generic exception-reporting methods that can be used with any JDBC program. This type of exception handling is required because often multiple exceptions (`SQLException`) are chained together and then thrown. A `while` loop is used to report on each error in the chain. The program starts this process by calling the `errorPrint` method from the `catch` block of the code that accesses the database.

```
//  Beginning of the primary catch block: uses errorPrint method
}  catch (Throwable e)  {
   /*    Catch all exceptions and pass them to
   **    the exception reporting method   */
   System.out.println(" . . . exception thrown:");
   errorPrint(e);
}
```

The `errorPrint` method prints a stack trace for all exceptions except a `SQLException`. Each `SQLException` is passed to the `SQLExceptionPrint` method.

```
static void errorPrint(Throwable e) {
   if (e instanceof SQLException)
      SQLExceptionPrint((SQLException)e);
   else {
      System.out.println("A non SQL error occured.");
      e.printStackTrace();
   }
}  // END errorPrint
```

The `SQLExceptionPrint` method iterates through each of the exceptions on the stack. For each error, the method displays the codes, message, and stacktrace.

```
//  Iterates through a stack of SQLExceptions
static void SQLExceptionPrint(SQLException sqle) {
   while (sqle != null) {
      System.out.println("\n---SQLException Caught---\n");
      System.out.println("SQLState:   " + (sqle).getSQLState());
      System.out.println("Severity: " + (sqle).getErrorCode());
      System.out.println("Message:  " + (sqle).getMessage());
      sqle.printStackTrace();
      sqle = sqle.getNextException();
   }
}  //  END SQLExceptionPrint
```

To see the output produced by this method, type a wish-list item with more than 32 characters, such as `I wish to see a Java program fail`.

# Activity 4: Create and run a JDBC program using the client driver and Network Server

This activity demonstrates the ease with which a program that embeds Derby can be modified for a client/server implementation that uses the Derby Network Server.

This activity assumes you have performed the preceding activities and have a working directory called `DERBYTUTOR`, and have copies of the program files from the `$DERBY_HOME/demo/programs/workingwithderby/` directory. A basic knowledge of the `WwdEmbedded.java` program and experience starting and connecting to the Network Server are helpful. You will need to use a text editor to create the `WwdClient.java` program.

You will create a Derby client program, `WwdClient.java`, by changing a few lines of the `WwdEmbedded.java` program. You can run the client program in multiple command shells, allowing simultaneous update from two or more sources.

You use two command windows (Server-Shell and Client-Shell) in this activity. You use the Server-Shell to start the Derby Network Server and display Network Server messages. You use the Client-Shell to edit, compile and run the newly created `WwdClient.java` program. You set the `CLASSPATH` environment variable in the Client-Shell to support the client JDBC program.

1. Create the `WwdClient` program using the following steps:
   a. Open a command window (Client-Shell).
   b. Change to the `DERBYTUTOR` directory.
   c. Make a copy of the `WwdEmbedded.java` program called `WwdClient.java`.

| Operating System | Command |
| --- | --- |
| UNIX (Korn Shell) | **`cp WwdEmbedded.java WwdClient.java`** |
| Windows | **`copy WwdEmbedded.java WwdClient.java`** |

   d. Open the `WwdClient.java` file in a text editor and update the class name to reflect the new file name:

```
Original declaration
     public class WwdEmbedded

New declaration
     public class WwdClient
```

   e. Edit the *DEFINE VARIABLES SECTION* of the program so that the `driver` variable contains the name of the Derby client driver class and the `connectionURL` variable contains the hostname and port number of the Network Server.

```
Original definitions
     String driver = "org.apache.derby.jdbc.EmbeddedDriver";
     String dbName="jdbcDemoDB";
     String connectionURL = "jdbc:derby:" + dbName +
 ";create=true";

New definitions
     String driver = "org.apache.derby.jdbc.ClientDriver";
     ...
     String connectionURL = "jdbc:derby://localhost:1527/" +
 dbName + ";create=true";
```

   f. Compile the application.

```
javac WwdClient.java
```

A command prompt appears if the compilation is successful. The binary file `WwdClient.class` is created. If an error message appears, modify the line indicated so that it is identical to the example.

2. Set up the client/server environment using the following steps:
   a. Open a command window (Server-Shell).
   b. Change to the `DERBYTUTOR` directory.
   c. Start the Network Server:

| Operating System | Command |
| --- | --- |
| UNIX (Korn Shell) | `java -jar $DERBY_HOME/lib/derbyrun.jar server start`<br>`2009-05-08 17:24:16.031 GMT : Security manager installed using the Basic server security policy.`<br>`2009-05-08 17:24:16.665 GMT : Apache Derby Network Server - 10.5.1.1 - (764942) started and ready to accept connections on port 1527` |
| Windows | `java -jar %DERBY_HOME%\lib\derbyrun.jar server start`<br>`2009-05-08 17:24:16.031 GMT : Security manager installed using the Basic server security policy.`<br>`2009-05-08 17:24:16.665 GMT : Apache Derby Network Server - 10.5.1.1 - (764942) started and ready to accept connections on port 1527` |

3. Run the client program using the following steps:
   a. Return to the Client-Shell window.
   b. If not already set, set the `CLASSPATH` environment variable to include the location of the file `derbyclient.jar`:

| Operating System | Command |
| --- | --- |
| UNIX (Korn Shell) | `export CLASSPATH=$DERBY_HOME/lib/derbyclient.jar:.` |
| Windows | `set CLASSPATH=%DERBY_HOME%\lib\derbyclient.jar;.` |

**> Important:** Include the dot (`.`) at the end of the command so that your current working directory is included in the classpath.

   c. Run the program:

```
java WwdClient
org.apache.derby.jdbc.ClientDriver loaded.
Connected to database jdbcDemoDB
Enter wish-list item (enter exit to end):
a sunny day
 __________________________________________________________
On 2009-05-08 13:12:09.058 I wished for a peppermint stick
On 2009-05-08 13:12:21.28 I wished for a long vacation
On 2009-05-08 13:27:22.581 I wished for a sunny day
 __________________________________________________________
Enter wish-list item (enter exit to end):
a new car
 __________________________________________________________
On 2009-05-08 13:12:09.058 I wished for a peppermint stick
On 2009-05-08 13:12:21.28 I wished for a long vacation
On 2009-05-08 13:27:22.581 I wished for a sunny day
On 2009-05-08 13:27:42.105 I wished for a new car
 __________________________________________________________
```

```
Enter wish-list item (enter exit to end):
exit
Closed connection
Getting Started With Derby JDBC program ending.
```

4. Shut down the Network Server:

| Operating System | Command |
|---|---|
| UNIX (Korn Shell) | **java -jar $DERBY_HOME/lib/derbyrun.jar server shutdown**<br>2009-05-08 17:28:47.171 GMT : Apache Derby Network Server - 10.5.1.1 - (764942) shutdown |
| Windows | **java -jar %DERBY_HOME%\lib\derbyrun.jar server shutdown**<br>2009-05-08 17:28:47.171 GMT : Apache Derby Network Server - 10.5.1.1 - (764942) shutdown |

The server shutdown confirmation appears in both command windows.

**Activity notes**

In a client/server environment, the client program is often used from other computers on the network. Whenever a system accepts connections from other computers, there is a chance of abuse. To maintain security, the Derby Network Server defaults to accepting connections only from clients running on the local machine (`localhost`). Before this or any other Derby client program can access the Network Server from another machine, additional steps should be taken to secure the Network Server environment. Once secured, the Network Server can be safely configured to accept connections from other machines. Refer to the "Network Server security" and "Running the Network Server under the security manager" sections of the *Java DB Server and Administration Guide* for important information on securing the Network Server and enabling network connections.

With the Network Server started, you can run the client program simultaneously in multiple windows. To demonstrate this, open two command windows and perform the substeps of the "Run the client program" step in each window. Both clients will operate without a problem. In contrast, it would not be possible for a program that uses the embedded driver (for example, `WwdEmbedded.java`) to access the database until the database or the Network Server is shut down.

You may have noticed that the client program does not shut down the database. This is because the database is a shared resource in a client/server environment and, in most cases, should be shut down only when the Network Server is shut down. If multiple clients are accessing the database and one client shuts down the database, the remaining clients will encounter a failure the next time they attempt an SQL command.

Derby's two architectures have caused confusion for some new Derby users, who mistakenly think that embedded is a single-user configuration. This is not true. The embedded driver supports multiple simultaneous connections, performs locking, and provides performance, integrity, and recoverability. Any application that uses the embedded driver can open multiple Derby connections and then provide a means for multiple users to interact with the database on each connection. The Derby Network Server is an example of such an application.

## What next with Derby?

Congratulations on completing the activities in this tutorial. You now have experience with using Derby in both the embedded and client/server architectures.

With this basic knowledge, you are ready to begin using Derby to address your own specific needs. As your next step in learning about this lightweight and powerful tool, visit the Apache Derby Web site.

**Use this link:** **WorkingWithDerby Resources page**
**Browser URL**: `http://wiki.apache.org/db-derby/WorkingWithDerby`

**Activities summary**

We hope these activities have helped you understand the steps needed to create and access Derby databases. Although Derby is easy to set up and use, you will find that it has the features and reliability of much larger database systems. The examples presented here do not begin to scratch the surface of what can be done.

Please take a few moments to become familiar with the many online resources available to Derby users and developers by browsing the Derby Web site at Apache. Whether you are performing a general evaluation of Derby or have a specific need to address, the link to the WorkingWithDerby resources page is a good stepping stone to finding additional information of interest:

- The Derby Quick Start page is a good reference page organized by area of interest.
- You will find that many content-rich areas, such as the Derby Wiki and the Derby Users mailing list, are available to you.
- If you are interested in how others are using Derby, see the Uses of Derby page on the Wiki. This page contains informational links to development projects and products that use Derby. When you implement a system using Derby, please add it to this list.

# Quick start guide for experienced JDBC users

This section is for experienced JDBC programmers who already know how to set the classpath, how to run a Java program, and how to use a JDBC driver.

For more detailed information on the topics covered in this section, see the *Java DB Developer's Guide* and the *Java DB Tools and Utilities Guide*.

## Environments in which Derby can run

Before you configure your system to run Derby, it is useful to understand something about the different environments in which Derby can run, because these environments affect the classpath, driver name, and database connection URL.

See the *Java DB Developer's Guide* for more information on Derby environments.

### Embedded environment

An embedded environment is an environment in which only a single application can access a database at one time, and no network access occurs.

When an application starts an instance of Derby within its JVM, the application runs in an embedded environment. Loading the embedded driver starts Derby.

### Client/server environment

A client/server environment is an environment in which multiple applications connect to Derby over the network.

Derby runs embedded in a server framework that allows multiple network connections. (The framework itself starts an instance of Derby and runs in an embedded environment. However, the client applications do not run in the embedded environment.)

You can also embed Derby in any Java server framework.

See the *Java DB Server and Administration Guide* for more information on how to run Derby on a server.

## Available drivers

Different JDBC drivers are available depending on the environment you choose for Derby.

- org.apache.derby.jdbc.EmbeddedDriver

  A driver for embedded environments, when Derby runs in the same JVM as the application.
- org.apache.derby.jdbc.ClientDriver

  A driver for the Network Server environment. The Network Server sets up a client/server environment.

## Database connection URL

You must use a database connection URL when using the Derby-provided embedded driver to connect to a database.

The format for the database connection URL for connecting to a database is:

```
jdbc:derby:databaseName;URLAttributes
```

where:

- *databaseName*

  The name of the database that you want to connect to
- *URLAttributes*

  One or more of the supported attributes of the database connection URL, such as *;territory=ll_CC* or *;create=true*.

  For more information, see the *Java DB Developer's Guide*.

For the network client driver that is provided by Derby, the format for the database connection URL for connecting to a database is this:

```
jdbc:derby://server[:port]/databaseName[;URLAttributes=value[;...]]
```

where the *server* and *port* specify the host name (or IP address) and port number where the server is listening for requests and *databaseName* is the name of the database you want to connect to. The *URLAttributes* can be either Derby embedded or network client attributes. See the *Java DB Server and Administration Guide* for more information on accessing the Network Server by using the network client.

# Documentation conventions

This section describes the terminology, syntax, and typographical conventions of the Derby documentation.

## Terminology

The Derby documentation uses the specialized term `environment` to describe the method your application uses to interact with Derby.

An environment is sometimes referred to as a *framework*. The two types of environments are the *embedded environment* and the *client/server environment.*

## SQL syntax

SQL syntax is presented in modified BNF notation.

The meta-symbols of BNF are:

| Symbol | Meaning |
| --- | --- |
| \| | or. Choose one of the items |
| `[ ]` | Enclose optional items. |
| `*` | Flags items that you can repeat 0 or more times. Has a special meaning in some SQL statements. |
| `{ }` | Groups items so that they can be marked with one of the other symbols, i.e. [ ], \|, or *. |
| `( ) . ,` | Other punctuation that is part of the syntax. |

An example of how SQL syntax is presented:

```
CREATE [ UNIQUE ] INDEX IndexName
  ON TableName ( SimpleColumnName [ , SimpleColumnName ] * )
```

Command-line syntax for running Java programs and utilities (as well as examples) always begins with the word *java*:

```
java org.apache.derby.tools.ij
```

This documentation attempts to be JVM neutral, but any command line examples or syntax using JVM specific references should be verified with your JVM documentation.

## Typographical conventions

This documentation uses some typographical conventions to highlight elements of the SQL language, operating system commands, and the Java programming language.

**Table 1. Derby typographical conventions**

| Usage | Typeface | Examples |
| --- | --- | --- |
| New terms | Italic | defined by *keys* |

| Usage | Typeface | Examples |
| --- | --- | --- |
| File and directory names | Italic | *C:\derby* |
| Dictionary objects | Italic | The *Employees* table |
| In syntax, items that you do not type exactly as they appear, but replace with the appropriate name | Italic | `CREATE TABLE` *`tableName`* |
| SQL examples | Bold and/or fixed-width | **`SELECT city_name FROM Cities`** |
| Java application examples | Bold and/or fixed-width | `Connection conn = DriverManager.getConnection ("jdbc:derby:Sample")` |
| Things you type in a command prompt | Bold and/or fixed-width | **`java org.apache.derby.tools.ij`** |
| Comments within examples | Bold and/or fixed-width | `--This line ignored` |
| SQL keywords (commands) | All caps | You can use a CREATE TABLE statement |

# Derby libraries and scripts: complete reference

This section describes Derby libraries and scripts.

## Libraries provided by Derby

This section shows the different libraries used by Derby and their purposes.

**Engine library**

| Library Name | Use |
|---|---|
| derby.jar | For embedded databases.<br>You always need this library for embedded environments. For client/server environments, you only need this library on the server. |

**Tools libraries**

For embedded environments, you need a library in the classpath to use a tool. For a client/server environment, you need a library on the client only.

| Library Names | Use |
|---|---|
| derbytools.jar | Required for running all the Derby tools (such as `ij`, `dblook`, and import/export). |
| derbyrun.jar | Executable jar file that can be used to start the Derby tools. |

**The Network Server library**

| Library Name | Use |
|---|---|
| derbynet.jar | Required to start the Derby Network Server. |

**The network client library**

| Library Name | Use |
|---|---|
| derbyclient.jar | Required to use the Derby network client driver. |

**The locale libraries**

| Library Names | Use |
|---|---|
| • derbyLocale_cs.<br>• derbyLocale_de_<br>• derbyLocale_es.<br>• derbyLocale_fr.j<br>• derbyLocale_it.j<br>• derbyLocale_hu.<br>• derbyLocale_ja_<br>• derbyLocale_ko_<br>• derbyLocale_pl.j | Required to provide translated messages for the indicated locales. |

| | |
|---|---|
| • derbyLocale_pt_<br>• derbyLocale_ru.<br>• derbyLocale_zh_<br>• derbyLocale_zh_ | |

## Scripts included with Derby

Derby includes scripts that start the Derby tools, add the appropriate libraries to the CLASSPATH, and start and stop the Network Server. These scripts available only when you download the *bin* distribution of Derby.

The Derby scripts are located in the `/bin` directory where you installed Derby. To run the scripts, you need to set several environment variables so that the scripts run correctly. See Setting the environment variables to set the environment variables.

You must run the Derby scripts in an appropriate command line environment, such as a Windows command prompt or a UNIX shell.

The complete list of scripts that are included with Derby is:

***bin/dblook***
Runs the `dblook` tool.

***bin/ij***
Starts the `ij` tool.

***bin/NetworkServerControl***
Runs NetworkServerControl.

***bin/setEmbeddedCP***
Puts all of the Derby libraries for an embedded environment in the CLASSPATH.

***bin/setNetworkClientCP***
Puts the libraries needed to connect to the Derby Network Server into the CLASSPATH.

***bin/setNetworkServerCP***
Puts the libraries needed to start the Derby Network Server into the CLASSPATH.

***bin/startNetworkServer***
Starts the Network Server on the local machine.

***bin/stopNetworkServer***
Stops the Network Server on the local machine.

***bin/sysinfo***
Runs the `sysinfo` tool.

# Trademarks

The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the documents in the Apache Derby documentation library:

Cloudscape, DB2, DB2 Universal Database, DRDA, and IBM are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.